Algorithm for playing Hangman Game

Ramanan S

March 2022

1 Introduction

The task given here is to write an algorithm that plays the *Hangman* game. Hangman is a two player game where one person thinks of a word and another has to guess the word letter-by-letter. A player can make at most 5 wrong guesses and the game is over when either all the letters in the word are guessed or the player had made 6 wrong guesses.

2 Data Insights

The given data-set contains 227300 English words. Also, the data seems to contain a lot of words which are not part of the general English Vocabulary (Eg: 'aaa', 'zzz', etc). Average word length is 9.347. And maximum length of a word is 29. Most of the word's length lies between 5 to 11. In the given data, the most top-5 occurring alphabets are **e, i, a, r, n**. Around 90% of the words contains at least 2 of the letters from top-5 frequent alphabets. Hence, if we start with guessing the top-5 alphabets, then we would still have 3 tries left and the word will be partially filled.

2.1 Data cleaning

For removing noisy data (words not part of the English vocabulary), I tried the following method. I tried removing words in which any one of the alphabets occurs more than 3 times. After doing this, around 55K words were removed. But this didn't help the model as the winning percentage went down by 5%.

3 Algorithmic Model

For this problem, I plan to use a deep learning based approach. Transformer models are one of the popular choices for many of the NLP problems as they currently produce state-of-the-art results. The transformer model which we are going to use for this problem is **RoBERTa** . After experimenting with various versions of BERT, we found that the RoBERTa model performs the best.

The transformer models are trained by a method called **masked language modeling (MLM)**. In this training method, we take a sentence and mask some of the words, then we would train the model to predict those masked words. So, the problem setting for the hangman game is similar to the training method of the transformers.

3.1 About the model

BERT (Bidirectional Encoder Representations from Transformers) is a transformer model and RoBERTa is one of variants of BERT. BERT produced state-of-the-art results in various NLP tasks like Question answering, Language inference, etc,. BERT uses attention mechanism that learns relations between words in a sentence. Transformer architecture consists of two parts - an encoder and decoder with attention layers. This models reads the whole sequence of words at once and hence the training process can be parallelized, which is not possible in case of RNNs.

3.2 Training setting

For this problem, we first use a tokenizer which maps each alphabet to a vector. Then, we will mask some of the alphabets in each word and pass as input to the transformer model, the objective of the model is to predict those masked alphabets. On an average 35% of the total tokens have been masked and trained.

The data is split into two parts - train and test set. The transformer model was trained on 215,935 words (in batches of 16) and validated on 11,365 words.

3.3 Result

The model was trained for 3 epochs and validated on unseen data. The winning percentage was 51%. Then the model was used on practice runs, there the winning percentage was 51%. On the final run the model achieved 50% success rate.

3.4 Other possible approaches

- Using probability of letters occurring together (Bi-gram, Tri-gram probabilities).
- Using Recurrent Neural Networks. Given the current word (with masked alphabets), the RNNs can be trained to predict the next alphabet.